平成 13 年度

筑波大学第三学群情報学類

卒業研究論文

題目：３次元ビジュアルプログラミングシステム 3D-PP の編集効率の向上に関する研究

主専攻	情報科学
著者名	神田 正和
指導教員	電子・情報工学系 田中 二郎

要旨

我々は、大規模プログラムへの対応、リアリティの向上、レイアウト自由度の向上の点でビジュアルプログラミング環境の３次元化が有効と考え、３次元ビジュアルプログラミングシステム 3D-PP の開発を進めている。しかしながら、3D-PP には、プログラム編集操作の煩雑さに関する問題、自動レイアウトの再実行に関する問題、遠近法による３次元表現に関する問題、ノード選択に関する問題が存在する。

本論文では、この問題点を解決するために、ノードのグループ化、グループ化を用いたレイアウト保存、ノードの半透明表示を用いたノード隠蔽の回避、範囲指定による複数ノード選択手法を提案した。また、実際に 3D-PP に実装することで、3D-PP における編集操作を拡張し、3D-PP をより実用的な３次元ビジュアルプログラミングシステムに近づけた。

目次

第 1 章

はじめに

近年、計算機の急激な性能向上を背景として、ビジュアルプログラミングシステムの研究が多く行われている。ビジュアルプログラミングシステムとは、多くのプログラミング環境で用いられてきたテキスト表現のかわりに、アイコン、図形、アニメーションといったビジュアルな表現でプログラムを視覚化し、それをユーザが操作することでプログラミングするシステムである。プログラムが図形を用いて表されることにより、ユーザがより直感的に理解し、操作できるところに利点がある。

ビジュアルプログラミングシステムの多くは２次元表現を用いて、研究・開発されてきた [2, 3, 4]。我々は、ビジュアルプログラミングシステムに３次元表現を導入することにメリットがあると考え、３次元ビジュアルプログラミングシステム 3D-PP[6, 7, 8, 9, 10, 11, 12] の研究を進めている。

３次元化によるメリットを以下にまとめる。

取り扱うプログラム量の拡大：ビジュアルプログラミングは、量がかさばる傾向にあり、大規模プログラミングに対応できないという問題がある [5]。プログラムを３次元に配置すれば、一画面で扱えるプログラム量が増える。

リアリティの向上：我々の住む現実世界は３次元空間である。プログラミング環境も３次元空間で構築したほうが、よりリアルな表現が可能となり、図形がわかりやすくなる。

レイアウトの自由度の向上：２次元では、どんなに見やすくレイアウトしようとしても図形が交差したり重なったりすることが多々あるが、３次元空間では、自由度が一つ増えるため図形の重なりや交差を回避することができる。

しかしながら、３次元表現を用いることによって、一画面に扱えるプログラム量が増える反面、ユーザに３次元情報に対する認知的負荷がかかってしまうといった問題点もある。また、マウスを用いて行う「ノード生成」、「エッジによる結線」などの単純な操作を何度も繰り返す必要があり、プログラム編集操作が煩雑になりがちである。

本論文では、ビジュアルプログラミングシステムにおける編集効率を向上させる手法として、ノードのグループ化、グループ化を用いたレイアウト保存、ノードの半透

明表示を用いたオブジェクト隠蔽の回避、範囲指定による複数ノード選択の提案および実装について述べる。

論文の構成は以下の通りである。まず２章では 3D-PP の概要、特徴および問題点について述べる。次に、３章では、 3D-PP の問題点を解決する手法について述べる。４章では、拡張した 3D-PP の操作方法について述べる。５章で関連研究を述べた後、６章で結論を述べる。

第 2 章

3 次元ビジュアルプログラミングシステム 3D-PP の現在の編集環境

本章では、我々が開発している 3 次元ビジュアルプログラミングシステム 3D-PP を取り上げ、その概要、編集環境、問題点について述べる。

2.1 3D-PP の概要

3D-PP[6, 7, 8, 9, 10, 11, 12] は、並列論理型言語 KL1[1] を従来の 2 次元図形による視覚化ではなく、3 次元図形による視覚化を取り入れたビジュアルプログラミングシステムであり、並列論理型言語の特徴であるゴールのリダクション過程や論理変数の具体化に着目した表現方法を実装している。

3D-PP では、ユーザは 3 次元表現されたプログラム要素であるノード・エッジを組み合わせてグラフを作成することにより、プログラムを記述することができる（図 2.1）。

3D-PP の特徴として以下の点が挙げられる。

- マウスのみを用いた直接操作
  マウスで画面上のノードを直接操作 [18] することにより、3 次元空間内の情報と直接インタラクションすることができ、直観的にノードを操作することが可能である。

- 単一ウインドウ
  プログラミング用のウインドウと実行用のウインドウを 1 つにして、プログラムしたものそのものが動くことによって、ユーザが頭の中でプログラムウインドウと実行ウインドウの間で情報を結びつける必要がなくなる。

- 自動レイアウト
  自動レイアウト [7, 19] とは、ノードが重なったり、エッジが交差したりすることを防ぐためにノードを自動配置してくれるシステムのことである。これにより、プログラムが見た目にわかりやすくなり、ユーザが混乱するのを防ぐ。

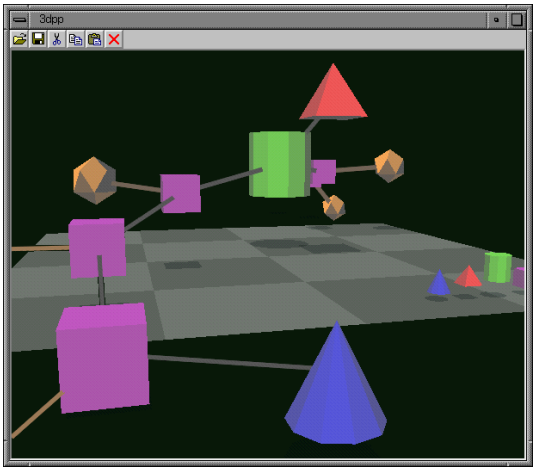

図 2.1: 3D-PP 実行画面

2.2 3D-PP における現在の編集環境

3D-PP における現在のマウスによる編集操作とメニューによる編集について以下にまとめる。

2.2.1 ３つボタンのマウスによる編集操作

3D-PP では、作業空間内に表示された３次元図形をマウスを用いて直接操作 [18] することで、３次元空間内の情報と直接インタラクションする。

ここでは、マウスを用いて行う編集操作について述べる。

- ノード生成
  作業空間内の３次元アイコン（図 2.2）を左ボタンクリックすることで、作業空間内にクリックした３次元アイコンと同じ形のノードを新しく生成することができる（図 2.3）。新しく生成されたノードは、クリックした３次元アイコンと同じ位置に生成される。また、３次元アイコンを左ボタンドラッグすることで、新しく生成したノードを操作対象とするノードの移動（後述）を行うことができる。

  なお、 3D-PP では、マウスカーソルを合わせたノードが振動することにより、ユーザが操作対象を認識するのを助けている。

- ３次元空間内のノードの移動
  ３次元空間内のノードをドラッグ＆ドロップすることで、ノードの移動を行うことができる。左ボタンによるドラッグ＆ドロップで、地面と平行な操作平面上で、地面との距離を変えることなくノードを移動させることができ（図 2.4）、右ボタンによるドラッグ＆ドロップで、視線と垂直な仮想平面上でノードを移

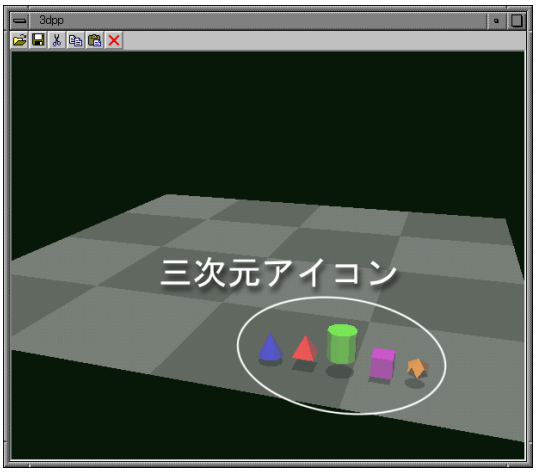

図 2.2: ３次元アイコン

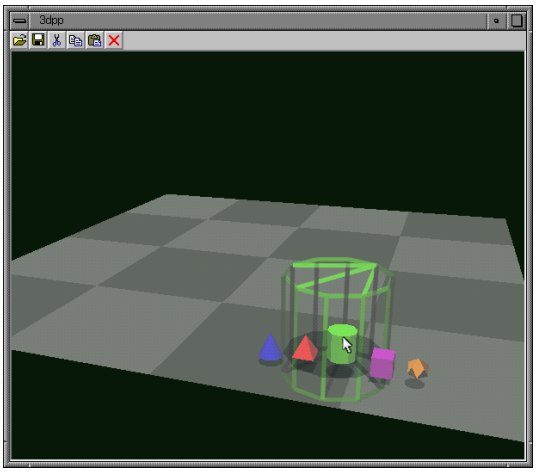

図 2.3: ノード生成

動させることができる（図 2.5）。なお、図 2.4、図 2.5では、操作平面を矢印で表している。また、移動中のノードはワイヤーフレーム表示となり、操作対象であることが分かるようになっており、かつ操作中のノードがその背後のノードを覆い隠して操作の障害とならないようにもなっている。

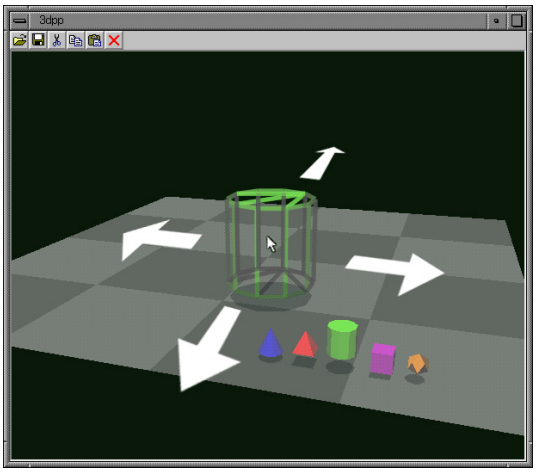

図 2.4: 左ボタンによる移動

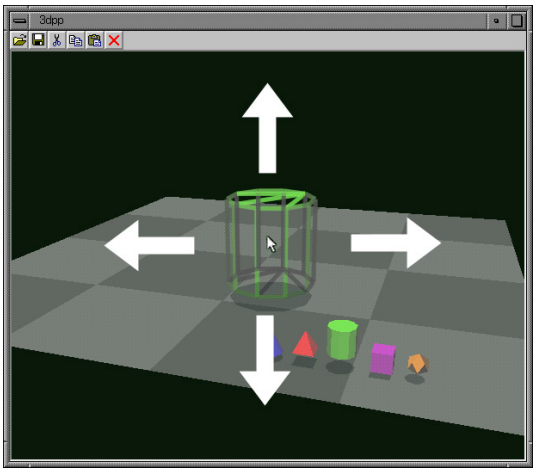

図 2.5: 右ボタンによる移動

- エッジ結線・結線解除
  エッジで結線したい２つのノードを、一方を始点、もう一方を終点として中ボタンドラッグ＆ドロップすることで、２つのノードがエッジで結線される（図 2.6, 図 2.7）。また、すでに２つのノードがエッジで結線されている場合は、エッジによる結線が解除される。

- ノード選択・選択解除
  ノードを左ボタンクリックすることでノードを選択状態にすることができる。

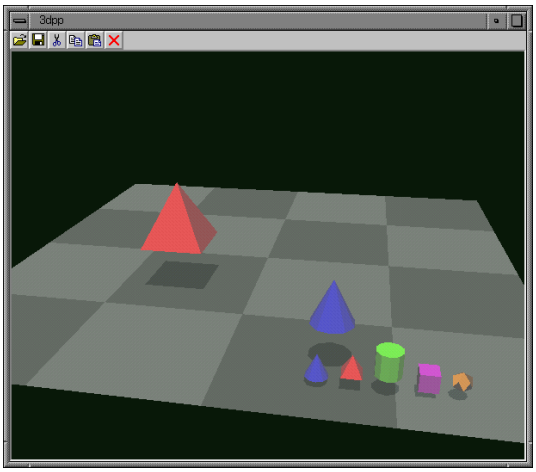

図 2.6: エッジ結線前

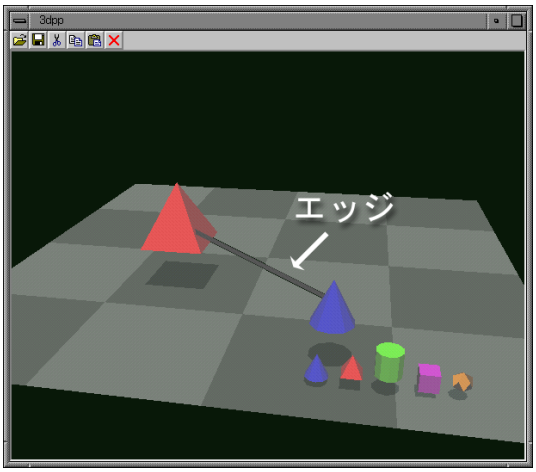

図 2.7: エッジ結線後

選択されたノードは半透明表示となり、他のノードと区別され、選択されていることがわかる。図 2.9では、緑色の円柱が選択されている。選択されたノードは、メニューによる編集の操作対象となる。また、選択されているノードを左ボタンクリックすることで選択状態を解除することができる。

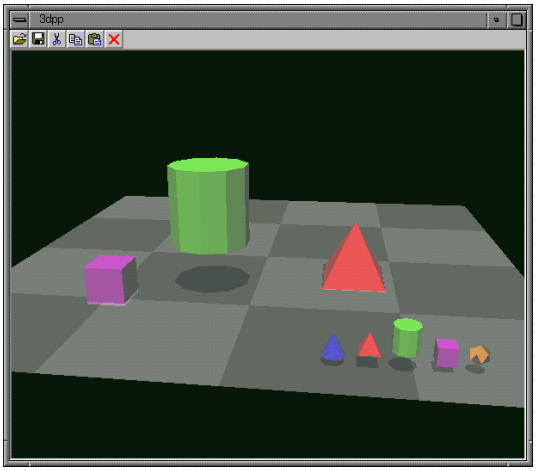

図 2.8: ノード選択前

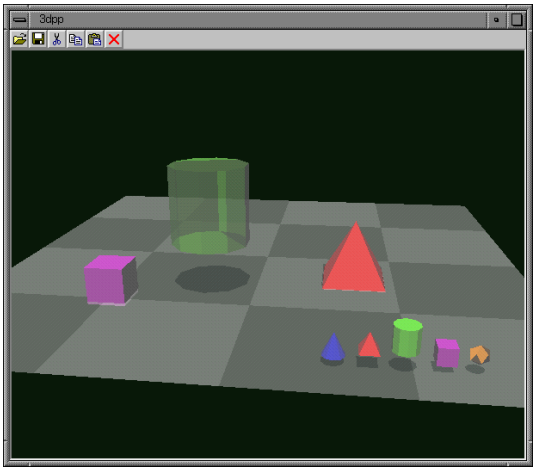

図 2.9: ノード選択後

- 視点変更
  ノード以外をドラッグ＆ドロップすることで、画面中央を中心として視点を変更することができる。なお、マウスの上下の動きに縦回転、左右の動きに横回転がそれぞれ対応している。

2.2.2 メニューによる編集

- Load , Save
  メニューバーの Load ボタンを押すとファイル選択ダイアログが出現するので、

そこで開きたいファイルを選択、またはファイル名を入力することで、以前に保存しておいたファイルからノード、エッジを読み込んで作業空間内に表示する。
また、メニューバーの Save ボタンを押すと、作業空間内のノード、エッジをファイルに保存することができる。

- Cut , Copy , Paste
  メニューバーの Cut、 Copy ボタンを押すことで、選択されているノードに対して、 Cut、 Copy を行う。 Cut または Copy を行った後、メニューバーの Paste ボタンを押すことで、 Paste が実行され、 Cut または Copy を行ったノードと同じ位置に新しくノードが現れる。

2.3 現在の 3D-PP の問題点

2.2節で述べた 3D-PP における現在の編集環境には、以下のような問題点がある。

2.3.1 プログラム編集操作の煩雑さに関する問題

ビジュアルプログラミングシステムでは、ノード数が増えるとユーザが把握しなければならない情報量が膨大となり、全体を把握するのが困難となる（図 2.10）。
また、プログラム全体を把握するためや、目的のノードを探すための視点変更操作を頻繁に行わなければならなくなるなど、ユーザが行わなければならない操作が複雑になり、編集効率も悪くなる。
特に、似た構造のプログラムを複数用いるときは、ノード生成やエッジ結線など単純な操作を何度も繰り返す必要があり、プログラム編集操作が煩雑になりがちである。

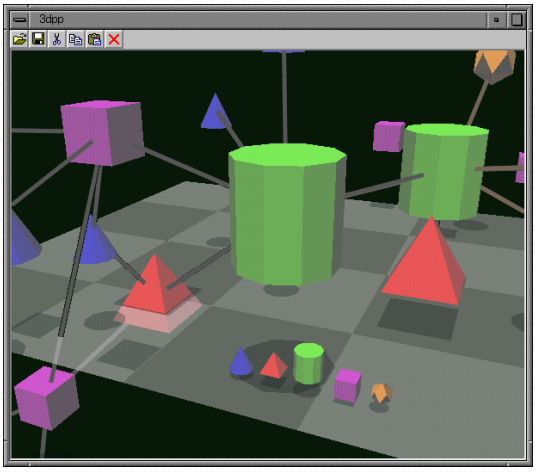

図 2.10: ノード数が増えた時

2.3.2　自動レイアウトの再実行に関する問題

従来のノード移動方法では、一つのノードを移動させると、そのノードが属しているグラフが自動レイアウト機能によって、移動させたノードの移動量だけ座標をずらして移動前と同じ形で再レイアウトされる（図 2.11）。これは、安定した状態のグラフを一度崩した後、自動レイアウトによって再び同じ形のグラフにするという無駄な処理を行っているということである。しかしながら、これでは自動レイアウトにかかる時間だけユーザが待つことになり、効率が良いとは言えない。

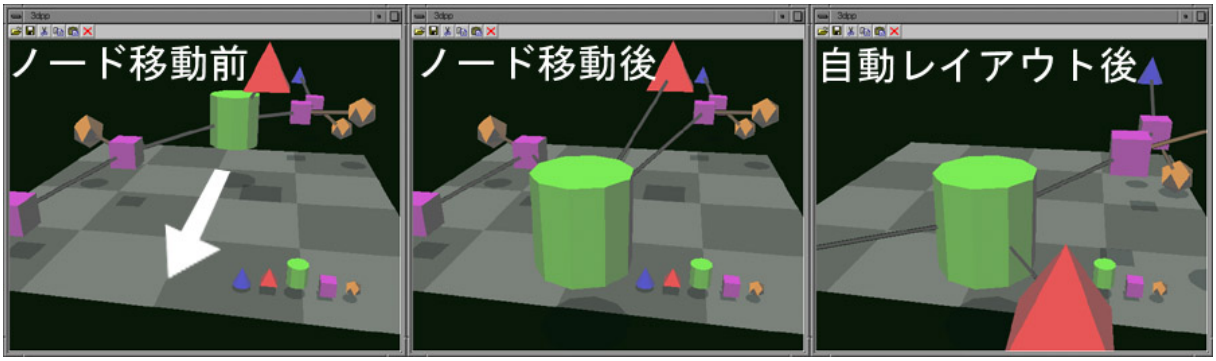

図 2.11: 自動レイアウトによるノードの再配置

2.3.3　遠近法による３次元表現に関する問題

３次元表現は遠近法を用いて描画されるため、ユーザのノード操作に障害を生じさせ、プログラムの全体把握を困難にしている。これは、３次元空間上でユーザの手前に表示されるノードが遠近法により大きく表示され、奥のノードが小さく表示されるからである。これにより、手前のノードが奥のノードを覆い隠してしまうことがある（図 2.12）。図 2.12では、緑色の円柱の後ろに、赤い四角錐と青い円錐があるにも関わらず、緑色の円柱に覆い隠されて見ることができない。

このように、手前のノードが奥のノードを覆い隠してしまうと、奥のノードを操作できないため、ユーザは奥のノードを確認・操作するために視点変更操作を行わなければならなくなる（図 2.13）。奥のノードを確認、または操作するためにわざわざ視点変更操作を行っていては、ユーザの行わなければならない操作が増え、編集効率が悪くなる。

特に手前のノードが、ドロップ対象などのように操作の対象となるノードを覆い隠してしまうと、ユーザは操作対象を確認できないうえに、操作もできないので非常に問題である。

2.3.4　ノード選択に関する問題

3D-PP では、一度の操作でノードを１つしか選択できない。そのため、複数のノードを選択するためには、選択したいノードの数の分だけノード選択操作を繰り返す必

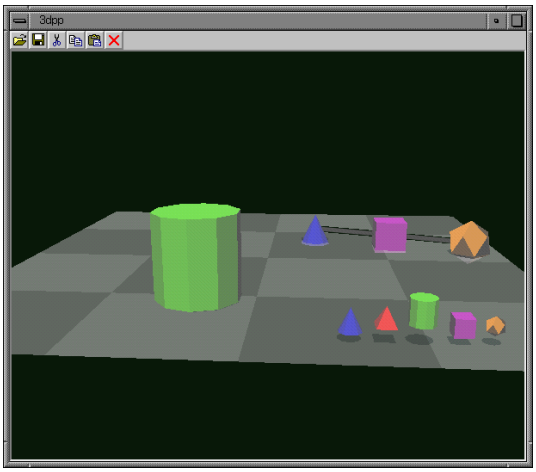

図 2.12: 遠近法による問題

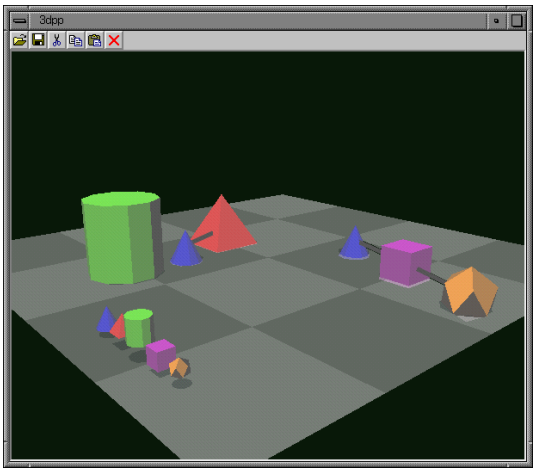

図 2.13: 別角度から見た図

要がある。これでは、選択したいノードの数が多い時などは、効率が良いとは言えない。

また、奥のノードを覆い隠している手前のノードを選択すると、手前のノードが半透明表示となり、覆い隠された奥のノードが見えるようになるので、あたかも奥のノードが操作できるように感じる。しかしながら、3D-PP では操作できないので、ユーザに誤解を与えてしまい、非常に問題である。また、見えているものを操作できないのは、直観的とは言えない。

第 3 章

3D-PP における編集環境の改善

本章では、前章で挙げた 3D-PP の問題点を解決し、ビジュアルプログラミングの効率を上げ、情報に対するユーザの認知的負荷を軽減する手法について説明する。

3.1 プログラム編集操作の煩雑さに関する問題の改善

ノードが増え、情報量が膨大となればなるほど、情報が複雑となり、ユーザに認知的負荷がかかる。情報が複雑なまま表示していては、ユーザは有用な情報を得ることが困難となり、プログラム編集も困難となる。そこで、グラフの適切な表示方法と操作法が必要となる。

ここでは、複数のノードを一つの操作の対象とするためのグループ化と、グループ化を利用したプログラムの注目部分の強調および、グループ化によるカスタマイズアイコンを用いたプログラムの再利用について述べる。

3.1.1 ノードのグループ化の導入

ここでは、複数のノードに対して同じ操作を与えるための、ノード、エッジに対するグループ化について述べる。ノード、エッジをグループ化することにより、複数のノードを一つのまとまりとして扱えるようにするのが目的である。

3D-PP におけるグループを以下のように定義する。

- 3D-PP におけるグループの位置付け
  グループは、ビジュアルプログラム編集、つまり、グラフ編集を助けるためのもので、編集時にのみ有効とし、実行には影響を与えないものとする。

- すべてのノード、エッジはいずれかのグループに属する
  ノード、エッジを作成した時点で必ずグループに属する。ゆえに、属しているノードが一つだけというグループも存在する。なお、エッジだけのグループは存在しない。

- ノード、エッジが属するグループは一つである
  ノード、エッジは複数のグループに属すことはない。

また、ノードをグループ化するにあたり、以下のことを考慮した。

- 専用のウインドウを作らない
  グループ化用にウインドウを作ったり、GUI コンポーネントを追加しない。3D-PP の特徴である、“単一ウインドウ”のスタイルを崩さないことにより、ユーザがプログラムウインドウとグループ化用ウインドウの間で情報を結び付けなくて良くなる。

- ３次元表現を使う
  ユーザに違和感を与えないために画面に表示する際には、３次元表現（３次元アイコン）を用いて表す。これにより、従来と同じ感覚で操作が可能であるとユーザに認識させることができる。

- 特殊な操作法を用いない
  従来と同様に、グループアイコンにマウスのみを用いて操作を行えるようにする。マウスの３つのボタンによる操作が増えるだけなので、ユーザにそれほど負担がかかならいと考えられる。

  以下にグループアイコンに対するボタン操作を示す。

  左ボタン：
  カレントグループを変更するのに使う（図 3.1）。カレントグループとは、現在編集対象となっているグループのことであり、作成されたノード、エッジは、カレントグループに属することとなる。なお、グループ列の末尾がカレントグループの時は新しくグループを作成する（図 3.1）。ここでいうグループ列とは、グループが作成された順に並んでいる列のことである。また、選択状態のノードがある場合は、選択状態のノードが属するグループを変更する。

  中ボタン：
  カレントグループに属するノード、エッジの不透明度を下げるのに使う（後述）。

  右ボタン：
  カレントグループを変更するのに使う（図 3.1）。なお、選択状態のノードがある場合は、選択状態のノードが属するグループを変更する。グループ列の先頭がカレントグループの時は変更しない。

3.1.2　プログラムの注目部分の強調

プログラム編集しているとき、ユーザが注目している部分が必ず存在する。注目している部分を強調することで、より編集しやすいプログラム環境をユーザに提供できるはずである。

そこで、ここでは、注目していないグループの不透明度を下げる手法について述べる。注目していないグループの不透明度を下げ、注目していないグループが見えにくくなることで、注目しているグループを際立たせることができる。これにより、注目しているグループを編集しやすくなると考えられる。図 3.2に、不透明度の値によっ

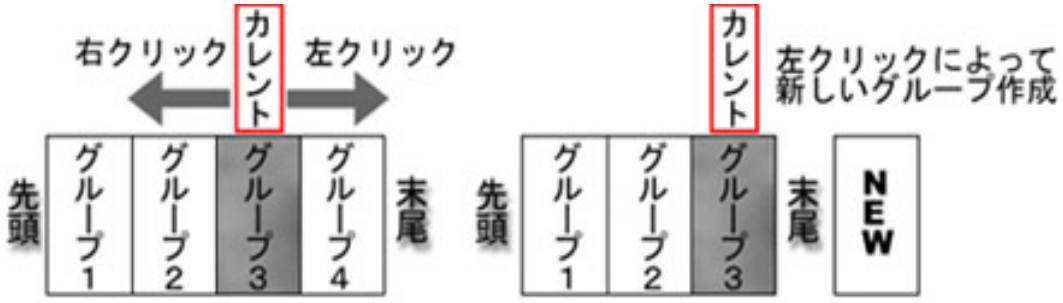

図 3.1: グループ列の操作

てノードがどのように見えるか示す。ここでは、緑色の円柱の不透明度を操作している。緑色の円柱の不透明度を下げるに従い、後ろに隠れていた赤の四角錐がはっきりと見えるようになる。

図 3.2: 不透明度の比較

この手法において以下のような点を考慮した。

- 注目しないグループをユーザが選ぶ
  システムが自動的に注目グループ以外の不透明度を下げるのではなく、ユーザが自由に不透明度を下げるグループを選ぶ。
  これにより、ユーザが必要だと思うグループだけに集中できる。なお、グループアイコンを中ボタンクリックすることによって、カレントグループに属するノード、エッジの不透明度を下げることにする。

- 注目していないグループのノードも操作可能
  注目していないグループのノードに対しても、従来と同じ操作が可能。
  不透明度を下げただけなので、ユーザは注目していないグループのノードも見ることができる。見えるノードに対して操作を行えるので直観的だと言える。
  また、注目していないグループのノードに対し操作をした際には、そのグループは注目されたみなし、不透明度を元に戻すことにする。

- 不透明度を変化させる際にはアニメーションを用いる
  不透明度を変化させるとき、一気に変化させないで、徐々に変化させる。また、その様子をアニメーションを用いて表す。これによって、ユーザが突然の変化に混乱するのを防ぐ。

3.1.3 プログラムの再利用

テキストベースのプログラミング環境では、似たようなコードを書く際に、コピー&ペーストによってプログラムを簡単に再利用することができる。しかしながら、3D-PP では複数のノードを選択するのが煩雑であり、コピー&ペーストによるプログラムの再利用は、面倒である。そこで、グループ化を用いたプログラムの再利用を提案する。

ユーザによってグループ化されたノードは、プログラム的にある程度意味をもっていると考える。プログラム的に意味があるということは、再利用する可能性が高いと考えられる。そこで、グループ化されたノードをファイルに保存し、カスタマイズアイコンとして利用することを考えた。カスタマイズアイコンから一度に複数のノードを生成することにより、プログラム編集効率が向上する。また、プログラムの再利用性が高まると思われる。また、このカスタマイズアイコンを、ユーザが自分の好きなように複数登録できるようにすることで、ユーザのプログラミングのパターンに対して対応できるようになる。

3.2 自動レイアウトの再実行に関する問題の改善

ノードの移動によって、レイアウトを一度崩した後、自動レイアウトによって再び同じ形のグラフにするという無駄な処理を行わないために、ここでは、グループ化を用いたレイアウト保存について述べる。

3.2.1 グループ化を用いたレイアウト保存

グループ化を用いたレイアウト保存の条件を以下のように定義する。

- グループ単位
  移動したノードが属するグループに含まれるノードをまとめて移動する。なお、同じグラフに属していても、グループが違えばまとめて移動しない。

- グラフ単位
  移動したノードが属するグラフをまとめて移動する。なお、同じグループに属していても、グラフが違えばまとめて移動しない。

以上の条件を付加したノード移動操作を 3D-PP に適用することにより、今まで再レイアウトにかかっていた時間をなくすことができる。

3.3　遠近法による３次元表現に関する問題の改善

３次元表現は遠近法を用いて描画されるため、手前のノードが奥のノードを覆い隠してしまうという問題があった。

ここでは、その問題点を解決する方法について説明する。

3.3.1　半透明表示を用いた隠蔽の回避

問題解決の方法として、手前のノードを半透明表示にすることを提案する。
これにより以下のような利点がある。

- プログラムの可読性の向上
  従来では視点変更操作を行わなければ見ることができなかった覆い隠された奥のノードが、手前のノードを半透明表示にすることで見えるようになり、プログラムの全体把握がしやすくなる。また、プログラムの可読性が向上すると考えられる。

- 覆い隠された遠くのノードが操作可能
  従来は、手前のノードしか操作できず、覆い隠されたノードを操作するためには、ユーザは視点変更操作をしなければならなかった。しかし、手前のノードを半透明表示にすることにより、従来では覆い隠されていて見えなかったノードを、ユーザが視点変更操作を行うことなく見ることができるようになる（図3.3）。さらに、従来では操作できなかった覆い隠されているノードも操作対象とすることにより、視点変更せずに覆い隠されているノードを操作でき、プログラム編集効率も上がる。

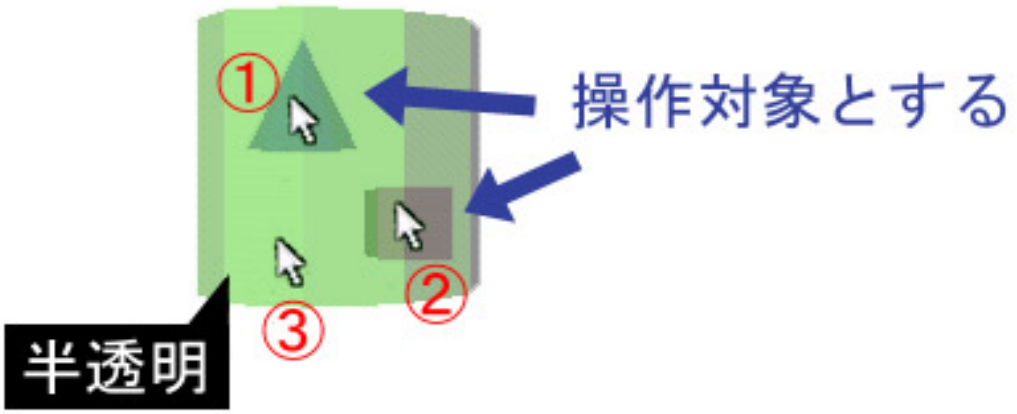

図 3.3: ノードの半透明表示

この手法において、以下の２つを考慮した。

- 奥のノードは半透明表示にしない
  手前のノードに覆い隠された奥のノードにマウスカーソルを合わせても半透明表示にしない。
  これは、奥にあるノードほど小さく表示されるので、階層が深くなるにつれて、操作対象としにくく、実用的でないからである。

- 操作対象を明らかにする
  図 3.3の①、②のように、奥のノードにマウスカーソルを合わせたときは、奥のノードを操作対象とする。このとき、マウスカーソルを合わせた奥のノードを振動させる。これにより、ユーザが操作対象を認識しやすくなる。

  また、③のように手前のノード以外触っていない場合は、手前のノード操作対象とする。

3.4　ノード選択に関する問題の改善

3D-PP では、一度の操作でノードを一つしか選択できなかった。しかし、これでは複数のノードを選択する際に操作が煩雑となってしまう。
そこで、ここでは、一度の操作でノードを複数選択する方法について説明する。

3.4.1　範囲指定によるノード選択

２次元のグラフィックツールなどでは、ドラッグにより範囲を指定して選択する方法が用いられている（図 3.4）。そこで、この範囲指定による選択方法を３次元用に拡張する方法を提案する。３次元用に拡張した３次元バウンディングボックスは、図 3.5のようになると考えられる。

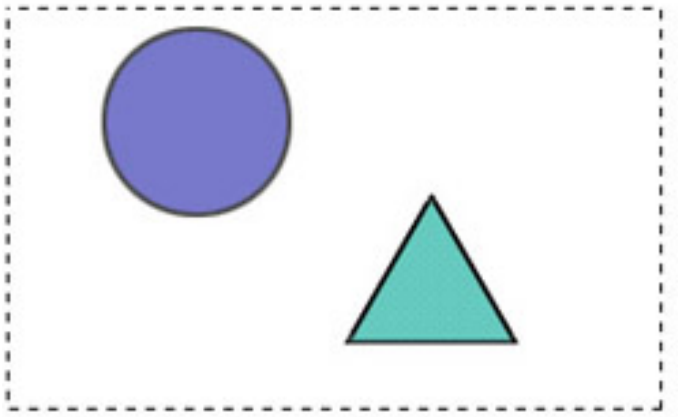

図 3.4: ２次元範囲選択

３次元用に拡張するにあたり、以下のことを考慮した。

- ２次元の範囲指定操作の感覚のまま行えるようにする
  ３次元であることを特に意識せず、あたかも２次元の範囲指定による選択を行っているような感覚で操作できるようにする。これは、３次元用の特別の操作を導入しないことによって可能だと考えられる。

- 選択状態のフィードバックを与える
  範囲指定したときに現れる３次元バウンディングボックスをドラッグの動きに合わせ、大きさが変わるようにする。また、ノードが３次元バウンディングボックスに含まれるときは選択状態にし、３次元バウンディングボックスの大きさが変わって、３次元バウンディングボックスに含まれなくなったときは、選択状態を解除するといった、フィードバックを与える。これにより、ユーザが範囲指定による選択操作を終えた際に、どのノードが選択されるか目に見てわかるようになる。

  なお、３次元バウンディングボックスに含まれているかどうかは、３次元図形の中心座標で判断することにする。

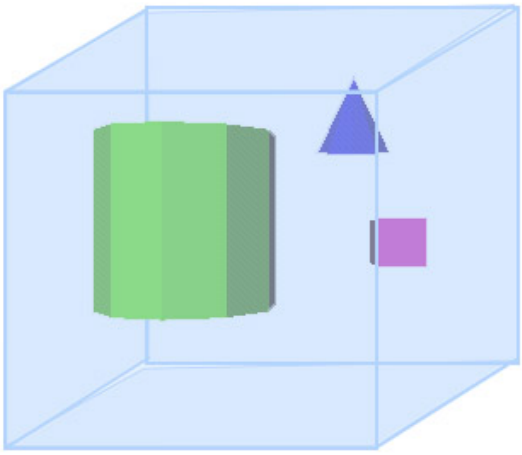

図 3.5: ３次元範囲選択

第 4 章

拡張した 3D-PP における編集操作

前章で述べた手法を我々が開発している 3 次元ビジュアルプログラミングシステム 3D-PP に実装し、プログラム編集環境の拡張を行うことで、ビジュアルプログラム編集の効率を向上させることができる。

4.1 拡張した 3D-PP の操作例

拡張した 3D-PP の新しい機能と操作について述べる。

4.1.1 ノードのグループ化

- ユーザによるグループ化
  ユーザが新しく生成したノードは、カレントグループに追加される。左下グループアイコン（図 4.1）を左ボタンクリックまたは右ボタンクリックでカレントグループを変更できる。なお、グループ列の末尾のグループがカレントの状態で左ボタンクリックすることで新しくグループを作成できる。また、地面を中ボタンクリックすることで、カレントグループのラベルを変更できる。

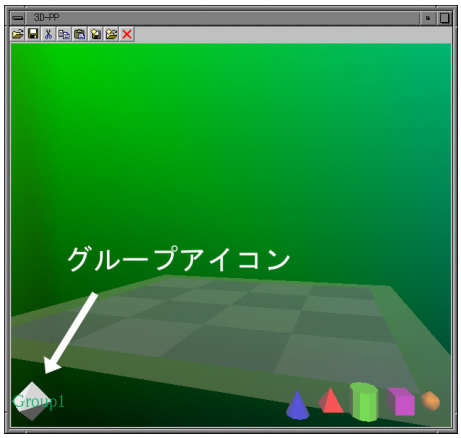

図 4.1: グループアイコン

- システムが行うグループ化
  ユーザがファイル読み出しを行った際に、そのファイルから読み出したノード、エッジは新しいグループとなる。また、グループのラベルはファイル名となる。

- 所属するグループの変更
  所属するグループを変更したいノードを選択状態にして、グループアイコンを左ボタンクリックまたは右ボタンクリックすることで所属するグループ変更できる。

4.1.2　カスタマイズアイコンを用いたプログラムの再利用

ここでは、グループ化を利用したカスタマイズアイコンの作成方法と作成したカスタマイズアイコンの利用方法および、カスタマイズアイコンを用いてプログラムを再利用する方法について説明する。

- カスタマイズアイコンの作成
  メニューバーのグループ保存アイコン（図 4.2）を押すと、カレントグループをにファイルに保存できる。このとき、保存するファイル名を指定するダイアログは表示されず、グループにつけられているラベルがファイル名となる。

- カスタマイズアイコンの登録と操作
  メニューバーのグループ読み出しアイコン（図 4.2）を押すと、ファイル選択ダイアログが現れる。ファイルを選択すると、カスタマイズアイコン（図 4.3）が画面に現れ、３次元アイコンとして新しく追加される。新しく追加されたカスタマイズアイコンを左ボタンクリックすることで、ファイル選択ダイアログで選択したファイルを読み込み、３次元空間内にノードが現れる。また、読み込まれたカスタマイズアイコンは保存され、次回起動時に同じ状態で復帰する。

図 4.2: メニューアイコン

図 4.3: カスタマイズアイコン

以下、カスタマイズアイコンを利用して構成要素数の多いリストを作成する方法を説明する。

1. Cons と Var をエッジで結線したカスタマイズアイコンを作成する
   まず、カスタマイズアイコンを作るために、 Cons と Var を用意する。次に、 Cons と Var をエッジで結線する。また、グループのラベルを cons に変更する（図 4.4）。この状態で、メニューバーのグループ保存アイコンを押すと、カスタマイズアイコン cons をファイルに保存できる。

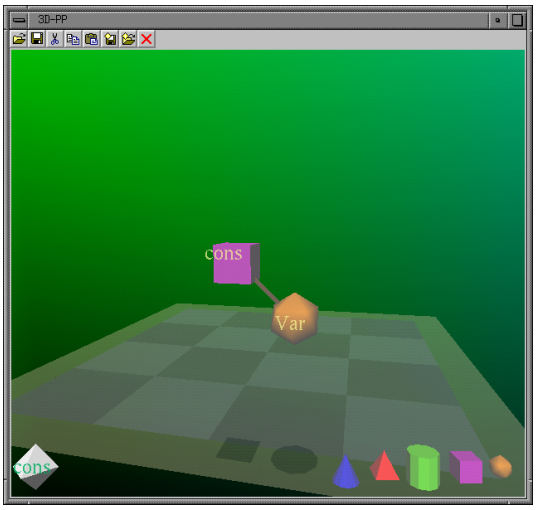

図 4.4: グループ cons

2. カスタマイズアイコンを読み込み、リストの構成要素を用意する
   次に、メニューバーのグループ読み出しアイコンを押して、カスタマイズアイコンを読み出す。このとき、ファイル選択ダイアログが現れるので、1 で保存した cons.grp を選択する。なお、この時読み出したカスタマイズアイコン cons は、画面左下のグループアイコンの上に表示される。この状態で、カスタマイズアイコン cons を左ボタンクリックする。これにより、 Cons と Var がエッジで結線された状態でカスタマイズアイコンの位置に出現する。ここで出現した Cons と Var は、同じグループ、同じグラフに属しているので、まとめて移動することができる。以下、同じ操作を繰り返して、リストに必要な数だけ用意する（図 4.5）。

3. リストの完成
   最後に、リストの構成要素である nil を用意して、図 4.6のようになるようにノード同士をエッジで結線し、ラベルを変更する。以上の操作を行うことにより、従来の方法に比べて格段に速く、リスト [a,b,c,d,e] を作成することができる。また、ユーザが行わなければならない操作の回数が減少するので、作業効率が向上した。なお、作成されたリストの構成要素である各ノードは同じグループ、

図 4.5: リストの構成要素

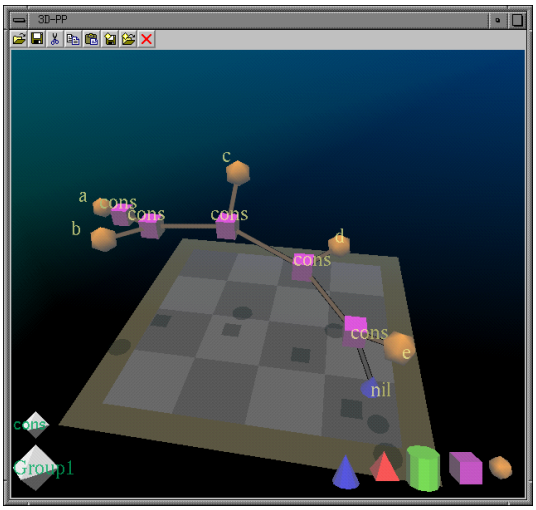

図 4.6: 作成されたリスト

同じグラフに属しているので、まとめて移動することができ、一つのリストだという感覚が強まる。

4.1.3 グループ化と不透明度操作を用いたプログラム編集

グループ化と透明度操作を用いたプログラム編集について、２つのリストを結合するプログラム append を作成する例を用いて説明する。 append を KL1 で記述すると以下のようになる。

```
:- module main.
main :- append([a,b,c],[e,d],Out),io::outstream([print(Out),nl]).

append([ ],In2,Out) :- Out = In2.
append([Msg|In1],In2,Out):-
        Out = [Msg|OutTail] , append(In1,In2,OutTail).
```

以下、この append プログラムの２行目の main の部分を、拡張した 3D-PP を用いて作成する例を順を追って説明する。

1. プロセス append の呼び出し
   まず、２つのリストを結合するプロセス append をファイルから呼び出す。この時、グループのラベルは append となる。また、２つのリストを結合した出力データを格納する Out を用意し、 append とエッジで結線する（図 4.7）。

2. リスト [a,b,c] の編集
   結合するためのリスト [a,b,c] を作成する。まず、グループアイコンを左ボタン

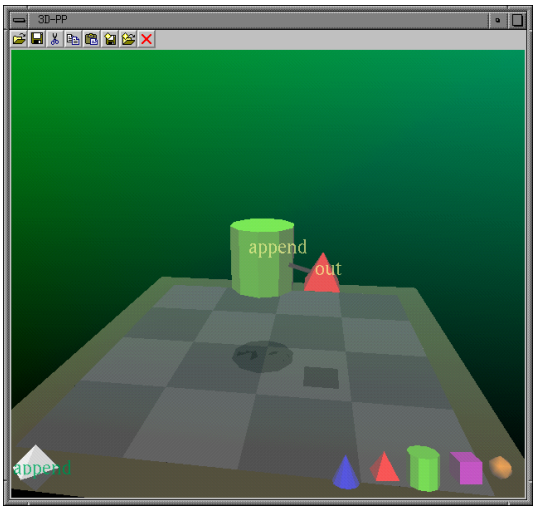

図 4.7: append と Out をエッジで結線したあと

クリックして新しくグループを作る。ここで、地面を中ボタンクリックして、新しく作成したグループのラベルを In1 に変更する。次に、リストの構成要素である、 a、 b、 c、 nil、 cons の各ノードを生成する。最後に図 4.8と同じになるようにノード同士をエッジで結線する。これで、 append の第一引数となるリスト [a,b,c] が出来上がった。

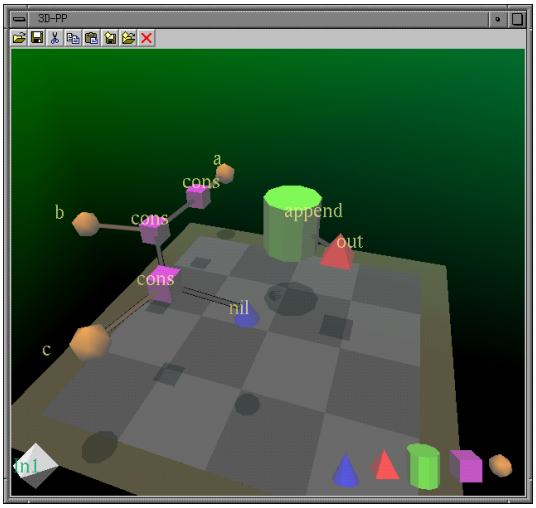

図 4.8: リスト [a,b,c]

3. リスト [a,b,c] の不透明度を下げる
   グループアイコンを中ボタンクリックして、グループ In1 に属するノードおよびエッジの不透明度を下げる（図 4.9）。このとき、グループ In1 に属するノードの影は、消えるようになっている。これにより、次に編集するリスト [d,e] に注目しやすくなる。

4. リスト [d,e] の編集
   リスト [a,b,c] と同様にリスト [d,e] を作成する。この時、グループのラベルは In2 とする。以上により、結合するための 2 つのリストが作成された（図 4.10）。最後に、グループ In1 の不透明度を元に戻し、2 つのリストと append をエッジで結線することにより、リスト [a,b,c] とリスト [d,e] を結合するビジュアルプログラムが作成できた（図 4.11）。

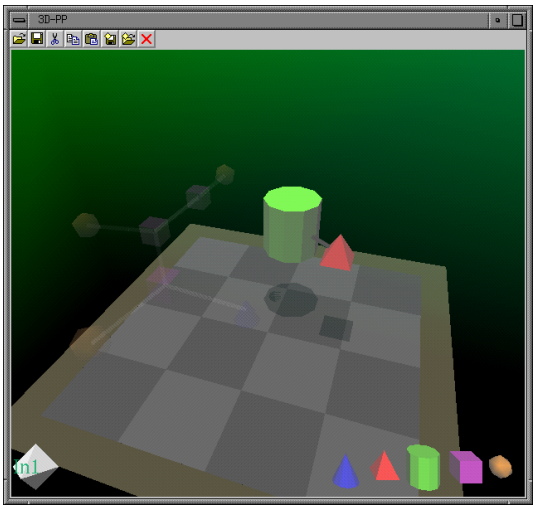

図 4.9: 不透明度を下げた後

図 4.10: リスト [d,e]

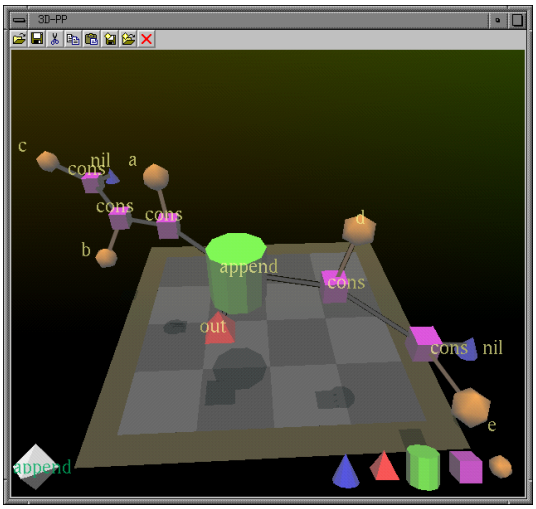

図 4.11: append を表すビジュアルプログラム

4.1.4 手前のノードに覆い隠された奥のノードの操作

手前のノードに隠れた奥のノードを、視点変更せずに操作する方法について順を追って説明する。

1. ノードにマウスカーソルを合わせる
   ノードにマウスカーソルを合わせると、マウスカーソルを合わせたノードが半透明表示になる。これによりマウスカーソルを合わせたノードに覆い隠された奥のノードを見ることができる（図 4.12）。

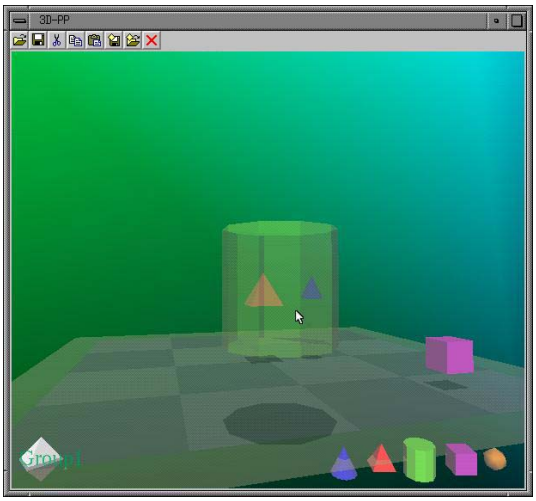

図 4.12: 半透明表示

2. 覆い隠されたノードにマウスカーソルを合わせる
   覆い隠された奥のノードにマウスカーソルを合わせると、マウスカーソルを合わせたノードが振動し、操作対象であることがわかる。

3. 覆い隠されたノードを操作する
   2 の状態で、覆い隠された奥のノードに対して移動操作を行うと、奥のノードが手前のノードに覆い隠されている間は手前のノードは半透明表示のままとなり、覆い隠された奥のノードを見て操作することができる。

4.1.5 範囲指定によるノード選択

一度の操作で複数のノードを選択する、範囲指定によるノード選択について順を追って説明する。

1. 始点を決め、ドラッグを開始する
   まず、ドラッグを開始する点を指定する。そして、中ボタンドラッグを開始する。ドラッグを開始すると、３次元バウンディングボックスが出現し、３次元バウンディングボックスの中に入っているノードが選択される（図 4.13）。このとき、クリックした画面の x,y 座標によって作業空間内の x,y 座標が決まり、作業空間内の z 座標はシステムが自動的に決める。なお、選択されたノードは、従来の 3D-PP とは違い、ワイヤーフレーム表示となる。これは、 4.1.4の操作で半透明表示を用いているため、選択状態を半透明表示にすると、ユーザが誤解してしまうからである。

2. 終点を決める

   ドラッグしている間、３次元バウンディングボックスの大きさがマウスの動きによって変わる。なお、３次元バウンディングボックスの大きさの変化によって、３次元バウンディングボックスの外に出たノードは選択状態ではなくなる。以上の操作を行い、目的のノードが３次元バウンディングボックスの中に入ったところで、ドラッグをやめる（図 4.14）。すると、３次元バウンディングボックスの中に入っているノードを選択して、３次元バウンディングボックスは消え、範囲指定によるノード選択が終了する。

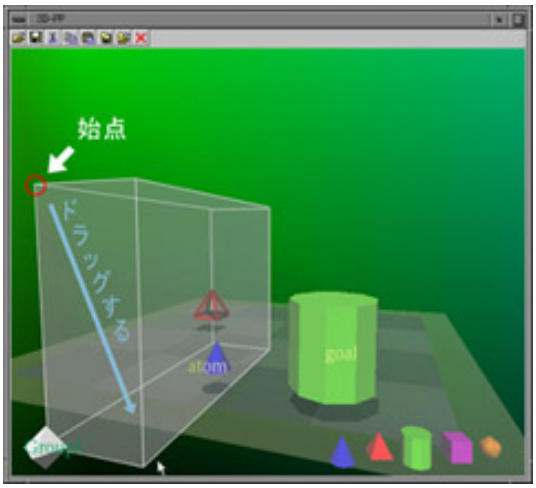

図 4.13: ドラッグ中の３次元バウンディングボックス

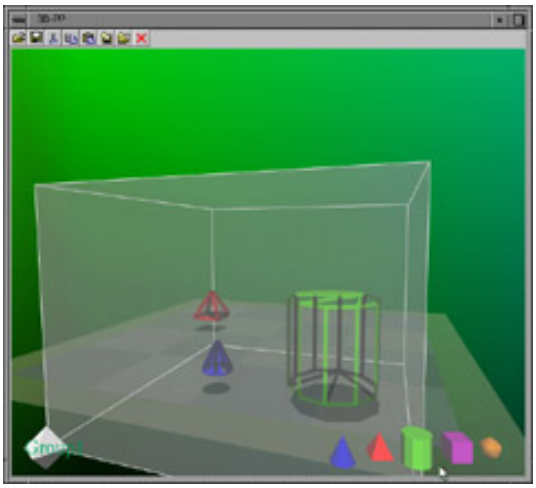

図 4.14: 終点決定時の３次元バウンディングボックス

4.2 開発環境

3D-PP は、 SGI O2(Irix 6.3) 上にて、汎用の３次元グラフィックライブラリである OpenGL と GUI ツールキットとして Qt を使用して実装されている。また、開発言語としては C++ を用いている。

第 5 章

関連研究

5.1 3D-Visulan

3D-Visulan[13, 14] は、ビットマップに基づくプログラミング言語 Visulan の 3 次元処理系である。ビットマップに基づくプログラミング言語とは、ビットマップ上にて、プログラム、データ、入出力などのコンピュータの全状態を表現し、そのビットマップが人とコンピュータとの対話空間となることで、絵を絵のまま扱うプログラミングを可能にする言語である。

3D-Visulan の操作インタフェースは、それまでの 2 次元のものを踏襲しており、特に 3 次元に利点のあるものではない。また、編集はツールバーのメニューを用いて行うため、ユーザにとって直観的ではない。

5.2 VisuaLinda

VisuaLinda[15, 16] は、並列言語 Linda のプログラム実行の状態を視覚化するビジュアルプログラミングシステムである。なお、 VisuaLinda が視覚化するのは、実行過程のみであり、プログラム編集過程は視覚化しない。

また、 VisuaLinda での視点変更操作は、 3 次元空間の x 軸,y 軸,z 軸にそれぞれ対応したスクロールバーを用いて行われる。スクロールバーを用いて視点を変更する操作は間接的であり、ユーザにとって直観的でない。

5.3 Information Cube

Information Cube[17] は、階層構造情報のための 3 次元視覚化技法として、半透明表示を用いている。階層が深くなるにつれて、透明度が低くなり、見えなくなる。グラフは比較的簡素に表現され、ユーザが理解しやすいグラフの 3 次元階層表現を可能としている。

本研究では、 3 次元表現の奥行き情報によるオブジェクトの隠蔽を防ぐ目的で、半透明表示を使っている。そのため、透明度を変化させずに、奥にあるものをより見や

すいようにしている点で異なっている。

第 6 章

まとめ

本研究では、我々が開発している 3 次元ビジュアルプログラミングシステム 3D-PP の問題点として、ノード数が増えた時にプログラムの可読性や編集効率が悪くなるという問題、自動レイアウトの再実行に関する問題、手前のノードが奥のノードを覆い隠してしまうという遠近法による 3 次元表現に関する問題、複数ノード選択操作が煩雑であるという問題を取り上げた。また、これらの問題を解決し、プログラム編集効率を向上させる手法として、ノードのグループ化、グループ化を用いたレイアウト保存、ノードの半透明表示を用いた隠蔽の回避、範囲指定による複数ノード選択を提案した。さらに、提案した手法を 3D-PP に実装することで、 3D-PP における編集操作を拡張し、 3D-PP をより実用的な 3 次元ビジュアルプログラミングシステムに近づけた。

謝辞

本研究を進めるにあたり、担当教官である田中二郎先生、助手である志築文太郎先生、三浦元喜先生からは、貴重なご指導を頂きました。心より感謝致します。

また、IPLAB の皆さんからは、ゼミ等を通じて多くのアドバイスを頂きました。特に VS グループのメンバーである、小川徹さん、飯塚和久さん、劉学軍さん、山田英仁さん、岡村寿幸さんからは大変有益なご意見を頂きました。ここに感謝の意を表します。

参考文献

[1] KLIC 講習会テキスト -KL1 言語編 -, 財団法人 新世代コンピュータ開発技術機構 作成, 財団法人 日本情報処理開発協会開発研究室 改訂, 1995.

[2] 南雲淳, 田中二郎：“viewPP”：グラフ構造とアニメーション表現に基づくプログラム実行の視覚化, 日本ソフトウェア科学会第１４回大会論文集, pp.17-20, 1997.

[3] Masashi Toyoda, Buntarou Shizuki, Shin Takahashi, Satoshi Matsuoka and Etsuya Shibayama： Supporting Design Patterns in a Visual Parallel Dataflow Programming Environment, Proc.1997 IEEE Symposium on Visual Languages, 1997.

[4] P.T.Cox,F.R.Giles and T.Pietrzykowski： Prograph： A Step towards Liberating Programming from Textual Conditioning, 1989 IEEE Workshop on Visual Languages,Rome, pp.150-156, 1989.

[5] Margaret Burnett et al： Scaling Up Visual Programming Languages, IEEE Computer, Vol.28 No.3, pp45-54, March, 1995.

[6] 大芝崇：直観的操作に基づく３次元モデリングツールと３次元ビジュアルプログラミングシステムの構築, 平成１１年度 筑波大学大学院修士課程工学研究科修士論文, 2000.

[7] 宮下貴史：三次元ビジュアルプログラム編集環境の構築, 平成１１年度 筑波大学大学院修士課程理工学研究科修士論文, 2000.

[8] 宮城幸司：三次元ビジュアルプログラミング環境の構築, 平成１０年度 筑波大学大学院修士課程理工学研究科修士論文, 1999.

[9] 宮城幸司, 大芝崇, 田中二郎：三次元ビジュアル・プログラミング・システム 3D-PP, 日本ソフトウェア科学会第１５回大会論文集, pp.125-128, 1998.

[10] Takashi Oshiba and Jiro Tanaka：“3D-PP”： Visual Programming System with Three-Dimensional Representation, In Proceeding of International Symposium on Future Software Technology(ISFST'99), pp.61-66, Nanjing, China, Octorber 27th to 29th,1999.

[11] 甲斐健太郎：３次元ビジュアルプログラミングシステムにおける仮想空間内でのマウスによる移動操作方法の研究, 平成１２年度 筑波大学第三学群情報学類卒業論文, 2001.

[12] 山田英仁：３次元表示上での自由な配置が可能なメニューの研究, 平成１２年度 筑波大学第三学群情報学類卒業論文, 2001.

[13] Kakuya Yamamoto： 3D-Visulan： A 3D Programming Language for 3D Applications, Proc. of Pacific Workshop on Distributed Multimedia Systems, The Hong Kong Univ.of Science and Technology(Hong Kong), pp.199-206, 1996.

[14] 山本格也：ビットマップ型言語におけるモジュール機能, 情報処理学会論文誌, Vol.38, No.12, pp.2544-2551, 1997.

[15] 高田哲司, 小池英樹：並列言語 Linda のプログラムの実行状態の視覚化, インタラクティブシステムとソフトウェア I, 日本ソフトウェア科学会 WISS'93, pp.9-16, 1993.

[16] Hideki Koike, Tetsuji Takada, Toshiyuki Masui： VisuaLinda： A Framework for Visualizing Parallel Linda Programs, In Proceeding of 1997 IEEE Symposium on Visual Languages(VL'97), pp.174-180, 1997.

[17] Jun Rekimoto and Mark Green： The Information Cube： Using Transparency in 3D Information Visualization, インタラクティブシステムとソフトウェア I, 日本ソフトウェア科学会 WISS'93, pp.125-132, 1993.

[18] Ben Shneiderman： Direct Manipulation: A Step Beyond Programming Languages, IEEE Computer, vol.16, No8, pp.55-69, 1983.

[19] 杉山公造：グラフ自動描画法とその応用, 計測自動制御学会, 1993.